CHAPTER 9

Strategies of Computer Worms

"Worm: n., A self-replicating program able to propagate itself across network, typically having a detrimental effect."

—Concise Oxford English Dictionary, Revised Tenth Edition

9.1 Introduction

This chapter discusses the generic (or at least "typical") structure of advanced computer worms and the common strategies that computer worms use to invade new target systems. Computer worms primarily replicate on networks, but they represent a subclass of computer viruses. Interestingly enough, even in security research communities, many people imply that computer worms are dramatically different from computer viruses. In fact, even within CARO (Computer Antivirus Researchers Organization), researchers do not share a common view about what exactly can be classified as a "worm." We wish to share a common view, but well, at least a few of us agree that all computer worms are ultimately viruses[1]. Let me explain.

The network-oriented infection strategy is indeed a primary difference between viruses and computer worms. Moreover, worms usually do not need to infect files but propagate as standalone programs. Additionally, several worms can take control of remote systems without any help from the users, usually exploiting a vulnerability or set of vulnerabilities. These usual characteristics of computer worms, however, do not always hold. Table 9.1 shows several well-known threats.

Table 9.1

Well-Known Computer Worms and Their Infection Methods

Name / Discovered	Type	Infection	Execution Method
WM/ShareFun February 1997	Microsoft Mail dependent mailer	Word 6 and 7 documents	By user
Win/RedTeam January 1998	Injects outgoing mail to Eudora mailboxes	Infects Windows NE files	By user
W32/Ska@m (Happy99 worm) January 1999	32-bit Windows mailer worm	Infects WSOCK32.DLL (by inserting a little hook function)	By user
W97M/Melissa@mm March 1999	Word 97 mass-mailer worm	Infects other Word 97 documents	By user
VBS/LoveLetter@mm[2] May 2000	Visual Basic Script mass-mailer worm	Overwrites other VBS files with itself	By user
W32/Nimda@mm September 2001	32-bit Windows mass-mailer worm	Infects 32-bit PE files	Exploits vulnerabilities to execute itself on target

Table 9.1 suggests that infection of file objects is a fairly common technique among early, successful computer worms. According to one of the worm definitions, a worm must be self-contained and spread whole, not depending on attaching itself to a host file. However, this definition does not mean that worms cannot act as file infector viruses in addition to network-based propagators.

Of course, many other worms, such as Morris[3], Slapper[4], CodeRed, Ramen, Cheese[5], Sadmind[6], and Blaster, do not have file infection strategies but simply infect new nodes over the network. Thus defense methods against worms must focus on the protection of the network and the network-connected node.

9.2 The Generic Structure of Computer Worms

Each computer worm has a few essential components, such as the target locator and the infection propagator modules, and a couple of other nonessential modules, such as the remote control, update interface, life-cycle manager, and payload routines.

9.2.1 Target Locator

To spread rapidly on the network, the worm needs to be able to find new targets. Most worms search your system to discover e-mail addresses and simply send copies of themselves to such addresses. This is convenient for attackers because corporations typically need to allow e-mail messages across the corporate firewalls, thereby allowing an easy penetration point for the worm.

Many worms deploy techniques to scan the network for nodes on the IP level and even "fingerprint" the remote system to check whether such a system might be vulnerable.

9.2.2 Infection Propagator

A very important component of the worm is the strategy the worm uses to transfer itself to a new node and get control on the remote system. Most worms assume that you have a certain kind of system, such as a Windows machine, and send you a worm compatible with such systems. For example, the author of the worm can use any script language, document format, and binary or in-memory injected code (or a combination of these) to attack your system. Typically, the attacker tricks the recipient into executing the worm based on social engineering techniques. However, more and more worms deploy several exploit modules to execute the worm automatically on the vulnerable remote system without the user's help.

Exploitation of vulnerabilities is the subject of Chapter 10, "Exploits, Vulnerabilities, and Buffer Overflow Attacks."

Note

Some mini-worms such as W32/Witty and W32/Slammer appear to combine the target locator (network scan) and infection propagator in a single function call. However, they still support distinct features: the generation of random IP addresses and the propagation of the worm body to new targets.

9.2.3 Remote Control and Update Interface

Another important component of a worm is remote control using a communication module. Without such a module, the worm's author cannot control the worm network by sending control messages to the worm copies. Such remote control can allow the attacker to use the worm as a DDoS (distributed denial of service) tool[7] on the zombie network against several unknown targets.

An update or plug-in interface is an important feature of advanced worms to update the worm's code on an already-compromised system. A common problem for the attacker is that after a system is compromised with a particular exploit, it often cannot be exploited again with the same one. Such a problem helps the attacker to avoid multiple infections of the same node, which could result in a crash. However, the intruder can find many other ways to avoid multiple infections.

The attacker is interested in changing the behavior of the worm and even sending new infection strategies to as many compromised nodes as possible. The quick introduction of new infection vectors is especially dangerous. For example, the intruder can use a single exploit during the first 24 hours of the outbreak and then introduce a set of others via the worm's update interface.

9.2.4 Life-Cycle Manager

Some worm writers prefer to run a version of a computer worm for a preset period of time. For instance, the W32/Welchia.A worm "committed suicide" in early 2004, and then the B variant of Welchia was released in late February of 2004 to run for three more months. On the other hand, many worms have bugs in their life-cycle manager component and continue to run without ever stopping. Furthermore, we

often encounter variants of computer worms that were patched by others to give the worm "endless" life.

Consider the statistics collected on an individual Welchia honeypot administered by Frederic Perriot between August 2003 and February 2004, shown in Figure 9.1. The sudden drop of Welchia is related to its life-cycle manager, which triggers the worm's self-killing routine.

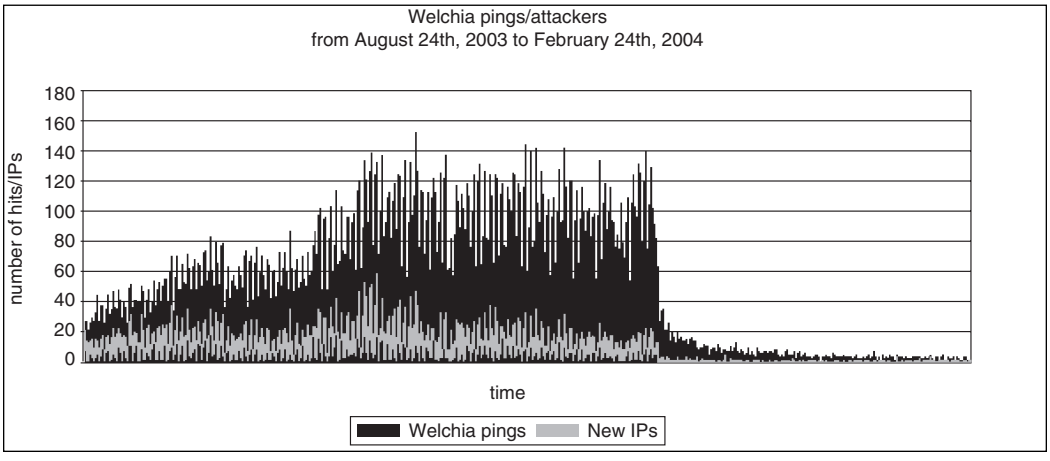

Figure 9.1 The suicide of Welchia worm.

The cumulative number of distinct Welchia attacking systems was around 30,000 when the worm started to kill itself when observed on a particular DSL network (see Figure 9.2).

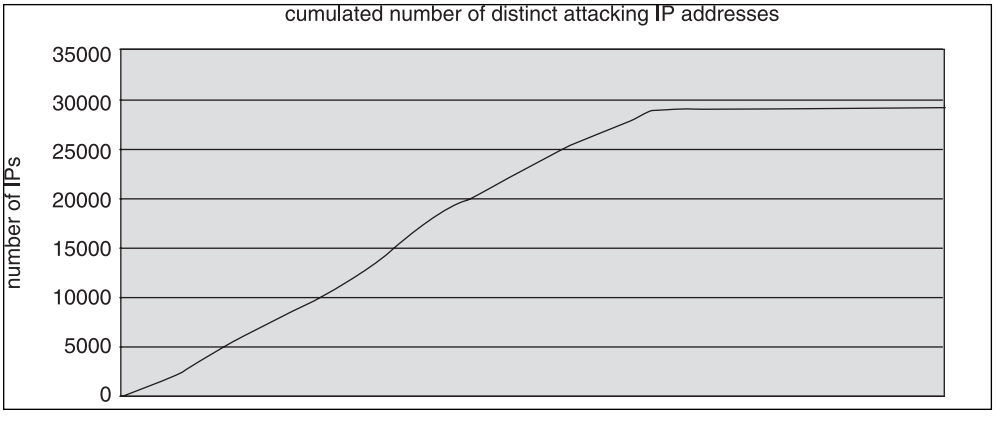

Figure 9.2 The cumulative number of Welchia attackers.

9.2.5 Payload

Another optional but common component of a computer worm is the payload (activation routine). In many cases, computer worms do not contain any payload. An increasingly popular payload is a DoS attack against a particular Web site. However, a common side effect of computer worms is accidental DoS attacks as a result of overloaded networks, especially overloaded network routers[8]. However, other interesting side effects have also been observed, such as accidental attacks on network printers.

Computer worms also can utilize the compromised systems as a "super computer." For example, W32/Opaserv[9] attempts to break a DES-like[10] secret key[11] by sharing the attack among the infected nodes, similarly to the SETI network. (In fact, some computer worms, such as W32/Hyd, download and install SETI to compromised systems. The W32/Bymer worm is an example of a DNETC [Distributed Network Client] installation to compromised systems.) Such attacks were first predicted in 1989[12].

Another interesting tendency is the planned interaction between two computer worms as a payload. Several antiworms have been released with the intention of killing other computer worms and installing patches against the vulnerabilities they exploited. Examples include Linux/Lion versus Linux/Cheese and W32/CodeRed versus W32/CodeGreen. In this chapter, I will also discuss other kinds of interactions between malicious programs.

Recently it is becoming popular to install an SMTP (Simple Mail Transfer Protocol) spam relay server as the payload of a worm. Spammers compromise systems on a large scale using worms such as W32/Bobax and then using the SMTP relay server created by the worm to spam messages from the "zombie" systems.

9.2.6 Self-Tracking

Many computer virus authors are interested in seeing how many machines the virus can infect. Alternatively, they want to allow others to track the path of the virus infections. Several viruses, such as W97M/Groov.A[13], upload the IP information of the infected system to an FTP site.

Computer worms typically send the attacker an e-mail message with information about the infected computer to track their spread. The Morris worm deployed a self-tracking module that attempted to send a UDP datagram to the host at ernie.berkeley.edu after approximately every 15 infections, but this routine was bogus, and it never sent any information[14]. A few other examples of self-tracking are mentioned later on in this chapter.

9.3 Target Locator

An efficient target locator module is an extremely important component of computer worms. The easiest mechanism for the attacker is to collect e-mail addresses on the system on which the worm was executed and to send attachments to such targets, but there are many more sophisticated techniques to reach new targets quickly, such as random construction of IP addresses in combination with port scanning.

Modern computer worms also attack the network using several protocols. In this section, I will summarize the most important attacks and network scanning techniques.

9.3.1 E-Mail Address Harvesting

There are many ways in which a computer worm can collect e-mail addresses for attacks. The attacker can enumerate various address books with standard APIs, including COM interfaces[15]. An example of this is W32/Serot[16].

Files can be enumerated directly to find e-mail addresses within them. Additionally, sophisticated worms might use the NNTP (network news transfer protocol) to read newsgroups or use search engines such as Google to collect e-mail addresses using techniques similar to those that spam attackers use.

9.3.1.1 Address-Book Worms

All computer environments have some form of address book to store contact information. For example, the Windows Address Book or the Outlook Address Book might contain the e-mail addresses of your friends, colleagues, and clients, or names of e-mail lists in which you participate. If a worm can query the e-mail addresses stored in such locations, it can send itself to all of them and spread with an exponential infection rate. Unfortunately, it is a rather trivial task to query the information in such address books.

The W97M/Melissa@mm[17] worm was especially successful with this technique in March 1999. The worm depends on the Microsoft Outlook installation on the system to propagate itself in e-mail by sending an infected Word document as an attachment.

9.3.1.2 File Parsing Attacks on the Disk

Several computer worms such as W32/Magistr[18] simply search for the e-mail client's files or for all files with a WAB extension and parse such files directly for

e-mail addresses. This technique became popular after Microsoft introduced security features in Outlook against computer worms that spread via e-mail messages.

As you might expect, file parsing–based attacks have their own minor caveats. For example, some worms have file format dependencies. The Windows Address Book is not saved in the same format on all Windows versions. Unicode is not always supported, and the file format is different in this case. This is why such worms cannot spread to other systems in such a situation. Problems like this can be extremely disturbing during natural infection tests in lab environments. It is an example of Murphy's Law when the whole world is infected with a particular worm—which fails to work in the lab environment.

Nevertheless, the technique seems to be efficient in the real world, and many successful worm attacks are the proof. For example, the W32/Mydoom@mm worm became extremely widespread in early 2004. Mydoom parsed files for e-mails with the following extensions: HTM, SHT, PHP, ASP, DBX, TBB, ADB, PL, WAB, and TXT.

Computer worms use heuristics to figure out whether a particular string is a possible e-mail address. One possible heuristic is to look for mailto: strings in HTML files and assume it is followed by an e-mail address. Occasionally, the size of the domain name is limited. For example, `somebody@a.com` might not be accepted by worms such as W32/Klez.H as a valid e-mail address, because "a.com" is too short to be good (although someone might configure a local network using such domain name). In addition, some worms target recipients with a specific language such as Hungarian and, to trick the user to execute the worm, they check the TLD (top-level domain) of e-mail addresses as suggested. For example, the Zafi.A worm sends itself to e-mail addresses that have ".hu" (Hungarian) as their TLD[19].

Sircam worm[20] searches for e-mail addresses in Internet Explorer's Cache directory, the user's Personal directory, and the directory that contains the Windows Address Books (referred to by HKCU\Software\Microsoft\WAB\WAB4\Wab File Name in the Registry) in files whose names begins with *sho*, *get*, or *hot*, or whose suffix is HTM or WAB.

9.3.1.3 NNTP-Based E-Mail Collectors

Attackers have long introduced their creations in Internet newsgroups. In 1996 the abuse of the News Net became very intense. As a result, researchers of the Dr. Solomon antivirus team decided to create a service called Virus Patrol[21] to scan Usenet messages for known and possibly unknown malware that was continuously planted in such messages. Virus Patrol was introduced in December 1996.

NNTP can be used in a number of malicious ways. For example, an attacker might be able to use a news server reader to build a large local database with the

e-mail addresses of millions of people. The attacker can use this database to help the initial fast propagation of the worm by running the worm on a system that hosts the database.

This is a common technique of spammers, and it is suspected that worms such as the W32/Sobig family were populated using such techniques. The newsgroup-based e-mail collector is not entirely unknown in Win32 viruses. In fact, the very first known Win32 virus that used e-mail to propagate itself used an NNTP collector. W32/Parvo[22] was introduced by the infamous virus writer GriYo of the 29A group in late 1998. Not surprisingly, just like many other GriYo viruses, Parvo also used polymorphism to infect PE files, but it also became the first virus to integrate an SMTP mass-mailing engine. Parvo was years ahead of its time, written in pure Assembly resulting in a 15KB virus body.

W32/Parvo used several newsgroups to collect e-mail addresses, but apparently a minor problem limited its spread. Parvo randomly tried to connect to two possible news servers: `talia.ibernet.es` or `diana.ibernet.es`. These servers, however, were not available to everyone at the time. Thus Parvo's newsgroup-based e-mail collector was limited to work "inside the borders" of Spain.

Parvo connects on port 119/TCP (NNTP) to one of the preceding servers and starts to communicate. The attacker prepared three different e-mail messages with content that he expected to be catchy enough for the selected audiences of three different newsgroups.

Parvo's first message targets frequent readers of hacking-related newsgroups, such as `alt.bio.hackers`, `alt.hacker`, `alt.hackers`, `alt.hackers.malicious`, and so on. The second message is sent to a subset of this newsgroup list. Finally, the third message targeted visitors to erotic newsgroups, such as `alt.binaries.erotica`, `alt.binaries.erotica.pornstar`, and so on.

To find e-mail addresses in newsgroups, Parvo uses the `group` command to join to a group randomly and then uses the `head` and `next` commands a random number of times to pick a message randomly. Finally, it extracts the e-mail address from the header of the randomly selected message, sends itself in e-mail to the target, and repeats the process.

9.3.1.4 E-Mail Address Harvesting on the Web

Attackers also can search for e-mail addresses using search engines. This is a relatively simple task that can help the attacker gain quick access to a large number of e-mails. As I was writing this book, the first such worms appeared that utilized popular search engines such as Google, Lycos, Yahoo!, and Altavista to harvest e-mail addresses. For example, the W32/Mydoom.M@mm worm used this technique

successfully, and according to Google, it caused minor DoS attacks against its servers.

9.3.1.5 E-Mail Address Harvesting via ICQ

Some computer worms, such as the polymorphic W32/Toal@mm[23], harvests e-mail addresses using ICQ (I Seek You) white pages located on ICQ servers. For example, `http://www.icq.com/whitepages/` allows you to make searches for contacts according to various characteristics such as name, nickname, gender, age, and country in any combinations and retrieve contact information, such as e-mail addresses, to people who meet your search criteria. Not surprisingly, computer worms can get an advantage of the information provided.

9.3.1.6 Monitoring User Access to SMTP and Newsgroups on the Fly

Alternatively, a computer worm can capture e-mail addresses from outgoing messages. Even if a particular e-mail address is not saved anywhere on the system, when the user sends a message to a particular address, the worm can send a message to the same address. The Happy99[24] worm was the first to use this method. Happy99 sends two messages that look similar to the example shown in Figure 9.3. Note the `X-Spanska: Yes` in the header. This is a self-tracking method that was used by the worm's author. SMTP servers simply ignore commands that begin with the "X" prefix.

```
Date: Fri, 26 Feb 1999 09:11:40 +0100 (CET)
From: "XYZ" <xyz@xyz.cz>
To: <samples@datafellows.com>
Subject: VIRUS
X-Spanska: Yes
```

Figure 9.3 The header section of an e-mail sent by Happy99.

(Message contains UU-encoded Attachment.)

The original message is shown in Figure 9.4.

```
From: "XYZ" <xyz@xyz.cz>
To: <samples@datafellows.com>
Subject: VIRUS
Date: Fri, 26 Feb 1999 09:13:51 +0100
X-MSMail-Priority: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3
```

Figure 9.4 The message of the user is also sent by Happy99.

The body of the extra mail contains a UU-encoded executable called happy99.exe. When the user executes the attached program, the worm's code is activated.

Happy99 looks for two API names in the WSOCK32.DLL export section. This DLL is the Windows Socket communication library used by many networked applications, including several popular e-mail clients. The worm patches the export address entries of the connect() and send() APIs to point to new entries at the end of the .text section (the slack space) of WSOCK32.DLL.

When the patched DLL is loaded in memory as a client library to a networked application, the worm intercepts the connect() and send() APIs. Whenever the user makes a connection, Happy99 checks the used ports. If the port turns out to be for mail or news access, a new DLL, SKA.DLL, is loaded into the process address space, which contains the worm's complete code previously saved on the disk.

When the intercepted send() API is called, the worm again checks whether this event is related to newsgroups or mail. If so, it copies some part of the original e-mail header, paying attention to MAIL FROM:, TO:, CC, BCC, and NEWSGROUPS: keywords in the header of the e-mail. Finally, it adds the X-Spanska: YES string to the mail header. Several other worms use an approach similar to Happy99's. Some of these worms inject their complete code into the WSOCK32 library.

9.3.1.7 Combined Methods

Of course, there can be many variations of e-mail address harvesting and worm propagation. For example, the Linux/Slapper worm[3] is capable of harvesting e-mail addresses and providing them to the attacker based on his request via a remote-control interface. Then another worm might be created by the attacker to use the database of harvested e-mail addresses to propagate to a large number of machines very rapidly—without requiring a large set of initial infections to harvest

an efficient number of e-mail addresses. Even more likely, the attacker can use the collected e-mail addresses to spam targets.

9.3.2 Network Share Enumeration Attacks

Probably the simplest method to find other nodes on the network quickly is to enumerate the network for remote systems. Windows systems are especially vulnerable to such attacks because of their rich support for finding other machines with simple interfaces. Computer viruses such as W32/Funlove used the enumeration principle to infect files on remote targets. These attacks caused major outbreaks at large corporations around the world.

Several computer worms have minor implementation problems and become overly successful at finding networked resources, including shared network printer resources. This happens because not all worms pay attention to the type of resources they enumerate, which can lead to accidental printing on the network printers. Indeed, bogus worms print random-looking binary garbage on the printer, which is in fact the code of the worm. W32/Bugbear and W32/Wangy are examples of computer worms that accidentally target network printers with such an attack.

The success of this kind of worm usually depends on the trusted relationship between systems. However, there are additional contributors:

- **Blank passwords:** Many default installations of systems are vulnerable to attacks because they do not have a default password set for administrative-level access on shared resources.
- **Weak passwords—dictionary attacks:** Weak passwords were a target of computer worms as early as 1988, starting with the Morris worm. However, password dictionary attacks on Windows systems did not become popular until 2003, with the sudden outbreak of worms like BAT/Mumu. Surprisingly, Mumu carried a relatively short password list that includes *password, passwd, admin, pass, 123, 1234, 12345, 123456,* and a blank password. Most likely, its success is related to the blank passwords on administrator accounts.
- **Vulnerabilities related to the handling of passwords:** The W32/Opaserv worm appeared in September of 2002 and became infamous for its attacks against systems that were otherwise protected with strong passwords, but that shared network resources on vulnerable Windows installations. Specifically, Opaserv exploited the vulnerability described in the MS00-072 security bulletin, which affected Microsoft Windows 95/98 and Me systems. This vulnerability, known as the share-level password vulnerability, allows

access to network shares using the first character of the password, no matter how long the password is. The number of systems that share network resources on the Internet without being protected by a personal firewall is overwhelming, which allows Opaserv easy access to writeable shared resources.

- **Password-capturing attacks to gain domain administrator-level rights:** In Windows networks, domain administrators have the right to read and write any files on any Windows machine on the network, unless specifically forbidden. On NT-based systems, domain administrators can also remotely execute programs on the fly and execute commands that require higher privilege levels than those of a regular user on the network.

These features make remote management possible, but at the same time they open up a whole new set of security problems. Gaining domain administrator rights is not trivial. However, a worm could do this easily if given enough time. A worm could spread through traditional channels, constantly sniffing the local network segment with traditional TCP/IP sniffing techniques. After detecting the domain administrator credentials being transferred in the network segment (for example, because the administrator is logging on from a nearby workstation), it logs the domain administrator's username and password hash.

NT-based networks do not broadcast the password in plain text; they run it through a one-way hash function first. The function cannot be reversed, so the password cannot be gathered directly from the hash. Instead, the worm could execute a brute-force attack to exhaust every possible password combination. It could run every password (A, AA, AAA, AAAA, and so on) through the same one-way function and compare the result. If they match, the password has been found. Alternatively, the worm could use a dictionary attack to find passwords as well.

With a strong password, this process might take days to accomplish, but a typical NT password takes less than a week to crack on a typical Windows workstation from a single Pentium system. Assuming that the worm could communicate with other compromised nodes, it could introduce workload balancing between the compromised nodes to share the work, making the cracking process even faster.

After the worm has cracked the NT domain administrator password, it owns the network and can do anything. Specifically, it can copy itself to any other Windows machine in the network. On NT-based machines, it can even start itself automatically with high access rights. Such a worm could also change the domain administrator password and the local administrator passwords to make itself more difficult to stop.

We first projected the feasibility of such attacks on NT domains with Mikko Hypponen back in 1997. At about the same time, tools such as L0phtCrack appeared to fulfill the sniffing and breaking of password hashes on NT domains. The authors of L0phtCrack demonstrated that long passwords can be often weaker than short ones when challenged with dictionary attacks[25].

In fact, the hashing algorithm of passwords on NT domains splits long passwords to seven character chunks, helping L0phtCrack crack the password more quickly. Nevertheless, computer worms with built-in network sniffing to crack passwords have not been discovered so far. Secure your passwords now—before it is too late! (Of course, this advice might not be funded very well when you consider a computer worm with a built-in keylogger to capture user accounts and passwords to attack other systems.)

9.3.3 Network Scanning and Target Fingerprinting

Several computer worms construct random IP addresses to attack other nodes on the network. By analyzing the scanning algorithm of the worm, someone might be able to make predictions about the worm's propagation speed on the network.

Evidently, an attacker can scan the entire Internet from a single machine, building IP addresses in a sequential manner (such as 3.1.1.1, 3.1.1.2, 3.1.1.3, and so on) and carefully ignoring invalid IP address ranges. This technique allows the attacker to build a "hit list" (database of IP addresses) to systems that might be vulnerable against a particular attack. To do that, the attacker typically fingerprints the remote systems just enough to suspect that the target may be vulnerable. In many cases, the fingerprinting is strongly related to a successful exploitation.

The hit list method is one of the theoretical backgrounds for so-called Warhol worms[26]. Warhol worms can infect 90% of all vulnerable systems on the entire Internet in less than 15 minutes. (It is expected that IPv6 will force computer worms to switch from traditional scanning methods to "hit list" techniques in the future.)

9.3.3.1 Scanning Using a Predefined Class Table: The Linux/Slapper Worm

Network worms can also scan for remote systems, generating random IP addresses but using a predefined table of network classes. For example, the Linux/Slapper worm uses the classes as defined in Listing 9.1 to attack possibly vulnerable Apache systems running on Linux:

Listing 9.1

The Class Definitions of the Linux/Slapper Worm

```
unsigned char classes[] = { 3, 4, 6, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 24, 25, 26, 28, 29, 30, 32, 33, 34, 35, 38, 40, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 61, 62, 63, 64, 65,
66, 67, 68, 80, 81, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153,
154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183,
184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 198, 199,
200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214,
215, 216, 217, 218, 219, 220, 224, 225, 226, 227, 228, 229, 230, 231, 232,
233, 234, 235, 236, 237, 238, 239 };
```

Note

I picked the name for Linux/Slapper worm when we discovered it in September 2002. I chose the name based on Slapper's similarity to the BSD/Scalper worm's code. The Scalper worm attacked Apache systems with the scalp exploit code—hence my name selection for this creature, after we had discovered it.

The preceding classes do not have some of the class A-sized, local networks, such as 10, or many other IP address ranges, including invalid classes. The worm builds the base IP address of the target machine as shown in Listing 9.2.

Listing 9.2

The Randomized IP Address Builder Routine of Linux/Slapper

```
a=classes[rand()%(sizeof classes)];
b=rand();
c=0;
d=0;
```

The attack will start with an address such as 199.8.0.0, and the worm will scan up the entire range of network nodes. Slapper attempts to connect on port 80 (HTTP) in order to fingerprint the remote system. It does so by sending a bogus HTTP request on port 80 that is missing the `Host:` header (which is required in HTTP/1.1) as shown in Listing 9.3.

continues

Listing 9.3

The Bogus GET Request of Linux/Slapper

```
GET / HTTP/1.1\r\n\r\n
```

The worm expects that Apache Web servers return an error message to this request; Apache returns the message shown in Listing 9.4 to the attacker node:

Listing 9.4

Apache Web Server's Answer

```
HTTP/1.1 400 Bad Request
Date: Mon, 23 Feb 2004 23:43:42 GMT
```

Listing 9.4 continued

Apache Web Server's Answer

```
Server: Apache/1.3.19 (UNIX)  (Red-Hat/Linux) mod_ssl/2.8.1
OpenSSL/0.9.6 DAV/1.0.2 PHP/4.0.4pl1 mod_perl/1.24_01
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

Note the `Server: Apache` keywords in the error message. The returned data also has information about the actual version number of the Web server, which is 1.3.19 in this example.

The worm checks whether the error message is coming from an Apache server by matching the server information. Then it uses a table filled with architecture and version information numbers (shown in Listing 9.5) to see if the target is compatible with the attack.

Listing 9.5

The Architectural Structure of Slapper

```
struct archs {
      char *os;
      char *apache;
      int func_addr;
} architectures[] = {
      {"Gentoo", "", 0x08086c34},
      {"Debian", "1.3.26", 0x080863cc},
      {"Red-Hat", "1.3.6", 0x080707ec},
      {"Red-Hat", "1.3.9", 0x0808ccc4},
      {"Red-Hat", "1.3.12", 0x0808f614},
      {"Red-Hat", "1.3.12", 0x0809251c},
      {"Red-Hat", "1.3.19", 0x0809af8c},
      {"Red-Hat", "1.3.20", 0x080994d4},
```

```
        {"Red-Hat", "1.3.26", 0x08161c14},
        {"Red-Hat", "1.3.23", 0x0808528c},
        {"Red-Hat", "1.3.22", 0x0808400c},
        {"SuSE", "1.3.12", 0x0809f54c},
        {"SuSE", "1.3.17", 0x08099984},
        {"SuSE", "1.3.19", 0x08099ec8},
        {"SuSE", "1.3.20", 0x08099da8},
        {"SuSE", "1.3.23", 0x08086168},
        {"SuSE", "1.3.23", 0x080861c8},
        {"Mandrake", "1.3.14", 0x0809d6c4},
        {"Mandrake", "1.3.19", 0x0809ea98},
        {"Mandrake", "1.3.20", 0x0809e97c},
        {"Mandrake", "1.3.23", 0x08086580},
        {"Slackware", "1.3.26", 0x083d37fc},
        {"Slackware", "1.3.26", 0x080b2100}
};
```

The attacker knows that the remote system runs Apache on a system that is likely to be compatible with the exploit code of the worm (assuming that the system is not patched yet). The third value is a "magic" address related to the exploit code. The magic number is explained in Chapter 10. In this example, the worm will select the 0x0809af8c address using the Red Hat and 1.3.19 architecture and version information. (See the bold line in the preceding structure.)

9.3.3.2 Randomized Scanning: The W32/Slammer Worm

So far, the Slammer worm has been responsible for the quickest worm outbreak in history. Slammer attacks UDP port 1434 (SQL server) and does not bother to check whether the IP address is valid. It simply generates completely random IP addresses and sends a packet to each target. (See Table 9.2 for an illustration.)

Table 9.2

A Sample Scan of the Slammer Worm

Time	Attacked IP Address:Port
0.00049448	186.63.210.15:1434
0.00110433	73.224.212.240:1434
0.00167424	156.250.31.226:1434
0.00227515	163.183.53.80:1434
0.00575352	142.92.63.3:1434
0.00600663	205.217.177.104:1434
0.00617341	16.30.92.25:1434
0.00633991	71.29.72.14:1434

continues

Table 9.2 continued

A Sample Scan of the Slammer Worm	
Time	**Attacked IP Address:Port**
0.00650697	162.187.243.220:1434
0.00667403	145.12.18.226:1434
0.00689780	196.149.3.211:1434
0.00706486	43.134.57.196:1434
0.00723192	246.16.168.21:1434
0.00734088	149.92.155.30:1434
0.00750710	184.181.180.134:1434
0.00767332	79.246.126.21:1434
0.00783926	138.80.13.228:1434
0.00800521	217.237.10.87:1434
0.00817112	236.17.200.51:1434

Slammer appears to be one of the quickest possible attacks on the Internet, but researchers predict that some worm types in the future will spread even faster. Slammer's infection was observed almost simultaneously all around the world and does not need to use any fingerprinting. It counts on the "sure shot" against vulnerable targets, which will continue the infection of other nodes as fireworks.

9.3.3.3 Combined Scanning Methods: The W32/Welchia Worm

The Welchia worm uses an IP address generator engine similar to Slapper's; however, it uses a combination of methods:

- Welchia scans class B–sized networks near the host's class-B network. It does so by scanning either the exact class B–sized network or slightly above or below, in hopes that such nearby systems also might be vulnerable to the same exploits.
- The worm uses a hit list for class A–sized networks. The attacker expects that these systems will have more vulnerable targets. This method also uses a randomized scanning strategy by attacking 65,536 random IP addresses.

Before Welchia proceeds with its exploits, it checks the availability of the remote system with ICMP echo requests (pings).

9.4 Infection Propagators

This section summarizes interesting techniques that computer worms use to propagate themselves to new systems.

9.4.1 Attacking Backdoor-Compromised Systems

Although most computer worms do not intentionally attack an already compromised system, some computer worms use other backdoor interfaces to propagate themselves. The W32/Borm worm was among the first computer worms to attack a backdoor-compromised remote system. W32/Borm cannot infect any other systems than those already compromised with Back Orifice (a fairly popular backdoor among attackers). Back Orifice supports a remote command interface that uses an encrypted channel between a client and the Back Orifice server installed on the compromised system. Borm utilizes a network-scanning and fingerprinting function to locate Back Orifice–compromised systems. See Figure 9.5 for illustration.

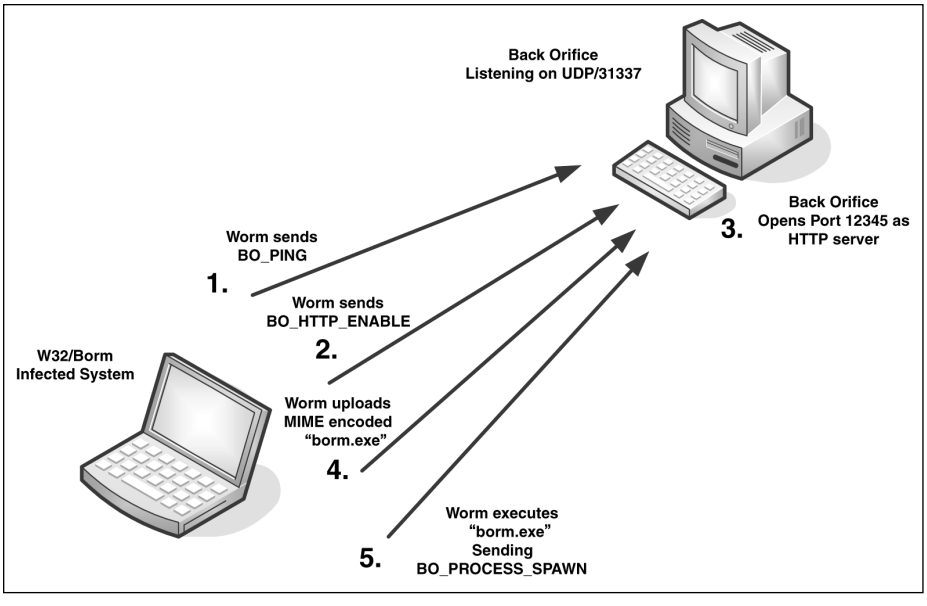

Figure 9.5 W32/Borm uses Back Orifice to propagate itself.

The worm attacks a compromised system using the following simple steps:

1. It randomly generates an IP address and actively scans using Back Orifice's BO_PING command to see whether the remote system is

compromised. To initiate any meaningful communication, the worm needs to know Back Orifice's magic password, which is **!*QWTY?*. The header of the communication data is encrypted with simple encryption, which is required by the Back Orifice server. Borm properly encrypts the data before it sends it to the randomly generated IP address on port 31337/UDP used by Back Orifice. If the remote node answers the BO_PING command, the worm proceeds to the next step. Otherwise, it generates other IP addresses to attack.

2. Borm sends a BO_HTTP_ENABLE command to the server.
3. In turn, this command instructs Back Orifice to create a virtual HTTP server on the compromised host. The worm instructs Back Orifice to use port 12345/TCP to establish an HTTP proxy on the compromised system.
4. Next, the worm connects and uploads itself in MIME-encoded format to the server.
5. Finally, the worm runs the uploaded executable on the server by sending a BO_PROCESS_SPAWN command. This will run the worm on the remote machine so it can start to scan for other Back Orifice systems from the newly infected node.

W32/Borm was among a flurry of computer worms that appeared in 2001. Other worms that utilized backdoor interfaces in this time frame included Nimda, which took advantage of a backdoor that was previously opened by a CodeRed II infection, and the W32/Leaves worm, which got in the wild by attacking SubSeven Trojan-compromised systems.

Borm was a creation of the Brazilian virus writer, Vecna. Several other of his methods are discussed later in this chapter.

9.4.2 Peer-to-Peer Network Attacks

Peer-to-peer attacks are an increasingly popular computer worm technique that not require advanced scanning methods on the computer network. Instead, such worms simply make a copy of themselves to a shared P2P folder on the disk. Anything that is available in a P2P download folder is searchable by other users on the P2P network.

In fact, some computer worms even create the shared folder in case the user only wants to search the P2P network for content instead of sharing similar content with others. Although this attack is similar to a Trojan installation, rather than recursive propagation, users of the P2P network will find the network-shared content easily and run the malicious code on their systems to complete the

infection cycle. Some P2P worms, such as W32/Maax, will infect files in the P2P folder. The most common infection technique is the overwriting method, but prepender and even appender infections have also been seen.

P2P clients such as KaZaA, KaZaA Lite, Limewire, Morpheus, Grokster, BearShare, and Edonkey among others are common targets of malicious code. P2P networks are an increasingly popular way to exchange digital music; they are hard to regulate because they are not centralized.

9.4.3 Instant Messaging Attacks

Instant messaging attacks originated in the abuse of the mIRC /DCC Send command. This command can be used to send a file to users connected to a particular discussion channel. Normally, attackers modify a local script file, such as script.ini used by mIRC to instruct the instant messaging client to send a file to a recipient any time a new participant joins a discussion.

Modern implementations of IRC (Internet Relay Chat) worms can connect dynamically to an IRC client and send messages that trick the recipient into executing a link or an attachment. In this way, the attacker can avoid modifying any local files.

For example, the W32/Choke worm uses the MSN Messenger API to send itself to other instant messaging participants as a "shooter game"[27]. Although several instant messenger software programs require the user to click a button to send a file, worms can enumerate the dialog boxes and "click" the button, so the actual user does not have to click. It is also expected that computer worms will exploit buffer overflow vulnerabilities in instant messenger software. For example, certain versions of AOL Instant Messenger software allow remote execution of arbitrary code via a long argument in a game request function[28].

9.4.4 E-Mail Worm Attacks and Deception Techniques

The vast majority of computer worms use e-mail to propagate themselves to other systems. It is interesting to see how attackers trick users every day, asking them to execute unknown code on their systems by sending them malicious code in e-mail. Let's face it: This is one of the greatest problems of security. How can security experts protect users from themselves?

During the last several years, an increasing number of users have been trapped in a "Matrix" of operating systems, such as Windows. Windows especially gives the illusion to millions of computer users worldwide that they are masters of their computers—not slaves to them. This illusion leads to neglected security practices.

In fact, most users do not know that they need to be careful with e-mail attachments. Consider W97M/Melissa, which used the following e-mail to trick recipients into executing the worm on their machines:

```
"Here is that document you asked for ... don't show anyone else ;-)"
```

Another common method of deception is forging e-mail headers. For example, the attacker might use the e-mail address of Microsoft's support as the W32/Parvo virus does, placing *support@microsoft.com* as a sender of the message. This can easily trick users into trusting an attachment and opening it without thinking. Other computers worms, such as W32/Hyd, wait until the user receives a message and quickly answers it by sending a copy of the worm back to the sender. Not surprisingly, this can be a very effective deception method.

Worms also make minor changes in the From: field to change the sender's e-mail randomly to something bogus. In practice, you might receive e-mail messages from many people, and most of the time they have nothing to do with the worm that abused their e-mail address. The bottom line is that notifying "the sender" will not necessarily help.

9.4.5 E-Mail Attachment Inserters

Some computer worms insert messages directly into the mailboxes of e-mail clients. In this way, the worm does not need to send the message; it simply relies on the e-mail client to send the mail. The earliest example of computer worms on Windows systems were of this type. An example of this is Win/Redteam, which targets outgoing mailboxes of the Eudora e-mail client.

9.4.6 SMTP Proxy–Based Attacks

W32/Taripox@mm[29] is an example of a tricky worm that acts as an SMTP (simple mail transfer protocol) proxy. This worm appeared in February of 2002. Taripox attacks the %WINDOWS %\SYSTEM32\DRIVERS\ETC\HOSTS file to proxy mail traffic to itself. Normally, the HOSTS file has a simple definition for the localhost address as shown in Listing 9.6.

Listing 9.6

The Content of a Typical Host's Configuration File

```
127.0.0.1       localhost
```

W32/Taripox remaps the IP address of the SMTP server to the local host. The worm can listen on port 25 (SMTP) and wait for any SMTP e-mail client to

connect. The outgoing e-mail message is then forwarded to the real SMTP server, but first the worm injects its own MIME-encoded attachment. The worm also tricks users with comment entries in the host file such as "# Leave this untouched," which is used for the localhost entry, and "# do not remove!," which is used as the comment for the SMTP IP address to localhost redirection entry. Figure 9.6 illustrates how Taripox works.

The HOSTS file is a common target for Retro worms to deny access to the Web sites of antivirus and security companies. Taripox's attack is similar to Happy99's, but it is a much simpler technique and does not require complicated modifications to binary files like WSOCK32.DLL.

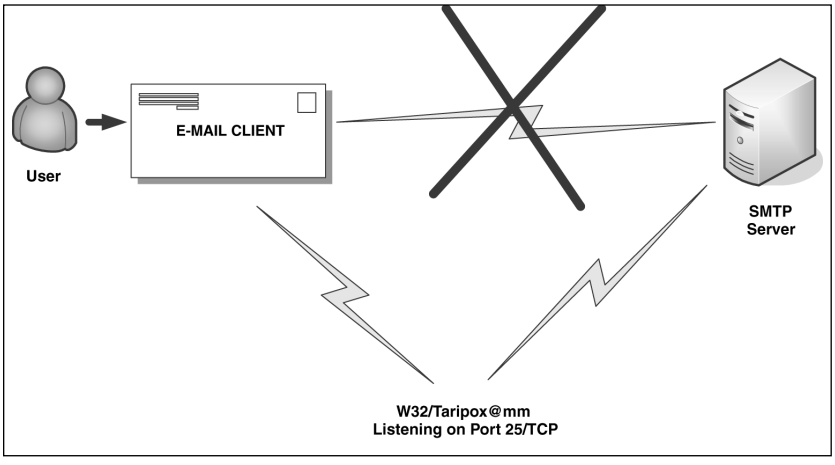

Figure 9.6 The W32/Taripox worm uses an SMTP proxy.

9.4.7 SMTP Attacks

As Microsoft strengthened Outlook's security to protect end users better against worm attacks, computer worm authors quickly started to use more and more SMTP-based attacks.

The first such major worldwide outbreak was caused by the Sircam[20] worm in July 2001 and was followed by the infamous W32/Nimda worm in September 2001. Smaller outbreaks had signaled the problem earlier, with the in-the-wild appearance of W32/ExploreZip in June 1999.

Sircam avoids relying on an e-mail program by getting the SMTP information directly from the Registry. This information consists of the following keys, which are created and used by a number of Microsoft mail applications:

- The current user's e-mail address: HKCU\Software\Microsoft\Internet Account Manager\Default Mail Account\Accounts\SMTP Email Address
- The address of the e-mail server: HKCU\Software\Microsoft\Internet Account Manager\Default Mail Account\Accounts\SMTP Server
- The user's display name: HKCU\Software\Microsoft\Internet Account Manager\Default Mail Account\Accounts\SMTP Display Name

If, for some reason, this information does not exist, Sircam will use `prodigy.net.mx` as the e-mail server, and the user's logon name as the e-mail address and display name. Using a hard-coded list of SMTP IP addresses is a common technique for computer worms, but usually this trick quickly overloads the particular set of servers once the worm is sufficiently widespread. Typically, such worms take off their SMTP servers with a DoS attack very quickly.

Thanks to implementation mistakes and bugs, it took a little while before SMTP worms could take their real place. Before Sircam, most worms lacked some important detail in their spreading mechanism. For instance, Magistr[18] often sends clean files or files that are infected but that reference some libraries not available on the recipient's system, so they fail to penetrate the target.

To illustrate the simplicity of SMTP, consider Table 9.3.

Table 9.3

A Typical SMTP Communication Between a Client and a Server	
1.) Client Connects to Server	
	2.) Server sends 220
3.) HELO name.com – *says hi to the server*	
	4.) Server sends 250
5.) MAIL FROM: <sender name>	
	6.) Server sends 250
7.) RCPT TO: <recipient name>	
	8.) Server sends 250
9.) DATA	
	10.) Server sends 354
11.) B*ody of the e-mail* Subject: <Any subject> (ENCODED ATTACHMENT FOLLOWS)	

. – dot to terminate

12.) Server sends 250

13.) QUIT - *to say goodbye to the server*

14.) Server sends 221

At this point, the e-mail might be dropped into a folder with a temporary name on the server in EML (Electronic Mail List) format. For example, Figure 9.7 shows an e-mail message sent by the W32/Aliz worm that was stored in the mail drop folder of Microsoft IIS Server.

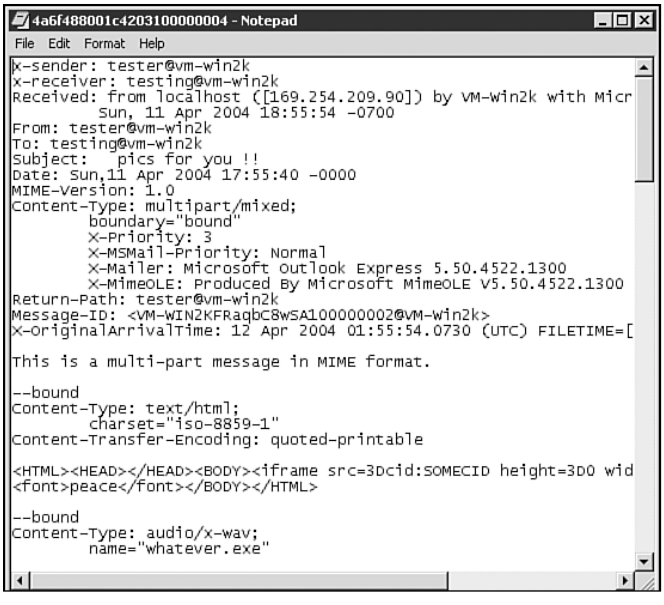

Figure 9.7 An e-mail message of the W32/Aliz worm.

This snippet already reveals the Content-Type exploit in the body of the e-mail, which will be discussed in more detail in Chapter 10. Chapter 15, "Malicious Code Analysis Techniques," also shows a network-level capture of W32/Aliz as an illustration of analysis techniques.

9.4.8 SMTP Propagation on Steroids Using MX Queries

Worms such as Nimda, Klez, Sobig, and Mydoom perfected SMTP mass-mailing by utilizing automated SMTP server address resolution using MX (eMail eXchanger) record lookups via the DNS (Domain Name System). Such worms check the domain name of the e-mail addresses they have harvested and obtain a valid SMTP server for that domain. Mydoom even uses the backup SMTP server addresses, instead of the primary server IP address, to reduce the load on SMTP servers further.

Table 9.4 is a list of MX look-ups of Mydoom on a test machine. The worm immediately sends itself to systems that it can find correctly, doing so many times per minute. In Table 9.4, the first column is the time in seconds, the second column is the infected system's IP address, and the third column shows the IP address of the DNS server used to make MX lookups.

Table 9.4

DNS Queries of the Mydoom Worm

Time	Workstation IP		DNS IP	Query Type	Queried Value
5.201889	192.168.0.1	->	192.168.0.3	DNS Standard query	MX dclf.npl.co.uk
5.450294	192.168.0.1	->	192.168.0.3	DNS Standard query	MX frec.bull.fr
6.651133	192.168.0.1	->	192.168.0.3	DNS Standard query	MX csc.liv.ac.uk
18.036855	192.168.0.1	->	192.168.0.3	DNS Standard query	MX esrf.fr
19.721399	192.168.0.1	->	192.168.0.3	DNS Standard query	MX welcom.gen.nz
30.761329	192.168.0.1	->	192.168.0.3	DNS Standard query	MX t-online.de
32.213049	192.168.0.1	->	192.168.0.3	DNS Standard query	MX welcom.gen.nz
32.447161	192.168.0.1	->	192.168.0.3	DNS Standard query	MX geocities.com

9.4.9 NNTP (Network News Transfer Protocol) Attacks

Worms such as Happy99 can mail themselves to newsgroups as well as to e-mail addresses. Usenet attacks can further enhance the spreadability of computer worms. Interestingly, most computer worms will only send mail directly to e-mail addresses.

9.5 Common Worm Code Transfer and Execution Techniques

Computer worms also differ how they propagate the worm's code from one system to another. Most computer worms simply propagate their main body as an attach-

ment in an e-mail. However, other types of worms utilize different methods, such as injected code and shellcode techniques in conjunction with exploit code, to attack another system.

9.5.1 Executable Code–Based Attacks

E-mail can be encoded in various ways, such as UU, BASE64 (MIME), and so on. However, the UU-encoded attachments are not very reliable over the Internet because UU uses some special characters whose interpretation depends on the context. Nowadays, most e-mail clients use MIME-encoded attachments by default—and that is how most e-mail worms' SMTP client engines transfer themselves to new targets. Script e-mail worms usually send attachments encoded according to the settings of the e-mail client on the compromised system.

9.5.2 Links to Web Sites or Web Proxies

Computer worms also can send links to executables hosted elsewhere, such as a single Web site, a set of Web sites, or an FTP location. The actual message on IRC or in e-mail might not have any malicious content in it directly—but infects indirectly. One problem with this kind of attack is the possibility of an accidental DoS attack against the system that hosts the worm's code. Another potential pitfall is that the defender can easily contact Internet service providers to request they disconnect such sites, preventing further propagation of the computer worm.

Tricky worms send links with the IP address of an already compromised system. First, the worm compromises a machine and opens a crude Web server on the system. Then it sends messages to other users, using the IP address of the machine with the port on which the worm itself is listening for a GET request. In this way, the worm attack becomes peer-to-peer, as Figure 9.8 illustrates. Such computer worms might be able to bypass content filtering easily if the content-filtering rule is based on attachment filtering.

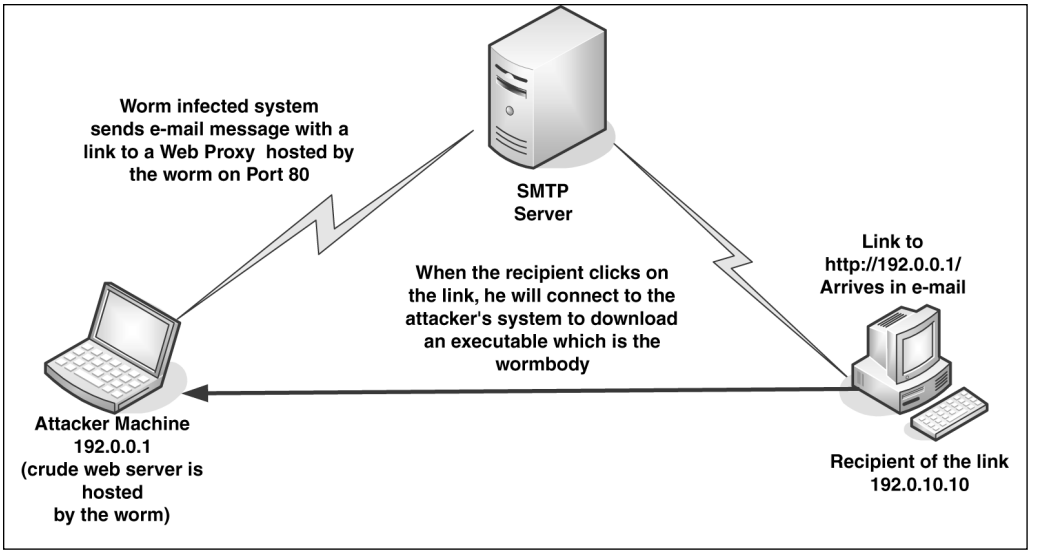

Figure 9.8 Tricky worms send links in e-mails instead of their own copy.

W32/Beagle.T used a similar method in March 2004. This variant of Beagle opens a crude Web server on TCP port 81. Then it sends a link to the recipient that triggers automated downloading with an HTML-based mail (which exploits the Microsoft Internet Explorer object tag vulnerability described in the *MS03-032* bulletin) to download and execute the hosted worm executable on the target automatically.

The W32/Aplore worm was among the first worms to use this attack to propagate itself on Instant Messaging in April 2002. When the W32/Aplore@mm[30] worm arrives on a new system, it acts as a crude local Web server on port 8180, hosting a Web page that instructs the user to download and run a program, which is the worm body itself. The worm tricks Instant Messenger users by sending them a link that looks like the following:

```
FREE PORN: http://free:porn@192.168.0.1:8180
```

where the IP address is the address of an infected system.

9.5.3 HTML-Based Mail

The e-mail can even be HTML mail-based. Disabling HTML support in your e-mail client reduces your chance of exposure to at least some of these threats, such as VBS/Bubbleboy. This worm is described in Chapter 10.

9.5.4 Remote Login–Based Attacks

On UNIX-like systems, commands such as rsh, rlogin, rcp, and rexec can be used directly by computer worms. Using such commands, worms can execute themselves on remote systems if the attacked system is not secured or if the password is guessed with a dictionary attack or similar method. Usually, such worms make a copy of their code directly to the remote system and execute themselves via the remote execution facilities.

On Windows systems, worms like JS/Spida can take advantage of vulnerable Microsoft SQL servers. Spida scans for remote Microsoft SQL server systems on port 1433 and tries to execute itself remotely with the following assumptions:

- The Microsoft SQL server runs in Administrative mode.
- The "sa" Microsoft SQL server account has no password set.

The worm takes advantage of the xp_cmdshell function to execute system commands to run the worm on the remote machine.

9.5.5 Code Injection Attacks

A more advanced attack requires exploitation of a target with direct code injection over the network. As traditional buffer overflows are getting more difficult to exploit, attackers are increasingly interested in exploiting server-side vulnerabilities related to a lack of input validation. For example, the Perl/Santy worm utilizes Google to find vulnerable Web sites and runs its own Perl script via a vulnerability in the phpBB bulletin board software. This worm successfully defaced tens of thousands of Web sites on December 21 of 2004. Depending on the thread model of the vulnerable target server, one of the following actions will happen:

- A new thread is created at the start of the server.
- A new thread is created upon each incoming request.

Furthermore, depending on the context of the hijacked thread, the worm

- Runs in SYSTEM context with high privileges.
- Runs in the context of a user with either high or low privileges that the worm might be able to escalate.

These preconditions are often reflected in the worm's operation. When, for example, W32/Slammer exploits a vulnerable Microsoft SQL server, the worm hijacks a thread that was executed at the start of the server. Thus the operations associated with the hijacked thread will be paralyzed because new incoming

requests will not be resolved. In addition, the server process and the entire system is heavily overloaded because the worm never stops sending itself to new targets.

An example of the second type of attack is W32/CodeRed. CodeRed exploits Microsoft IIS server via a malformed GET request. When the server receives the GET request, it executes a new thread to process it. The worm hijacks that particular thread and creates 100 new threads (300 in some variants) in the vulnerable server process. This kind of computer worm needs to avoid infecting the target a second time because the worm could exploit the target multiple times, causing the target to be overloaded shortly after the initial outbreak. In addition, computer worms that counterattack each other can also benefit from this condition because they can utilize the same exploit as their opponent.

Both of these attacks are detailed in Chapter 10 from the point of view of exploitation. Figure 9.9 illustrates this.

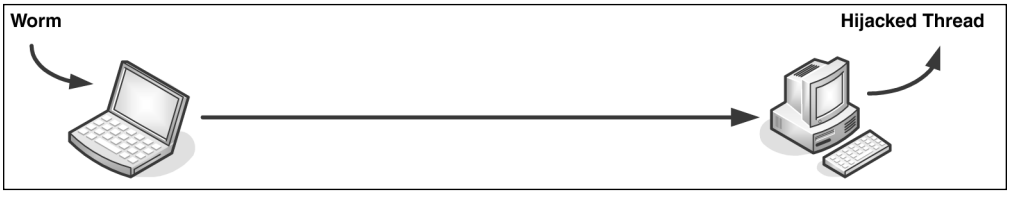

Figure 9.9 A typical one-way code injection attack.

In some cases, the injected code creates a new user account on the target that can be used by the attacker to log in to the system remotely.

Another interesting example of a code injection attack is the W32/Lespaul@mm worm. This worm takes advantage of a vulnerability in Eudora 5 that can be exploited by sending a malformed boundary tag.

Lespaul is a mass-mailer worm, but just like CodeRed or Slammer, it injects its code directly into the vulnerable Eudora 5 process. The worm does not send an attachment to the recipient; instead, it propagates itself as the mail body. It can appear in the Eudora mailbox as part of an e-mail message featuring an overly long header field; however, its code is never saved into a standalone executable at any point in order to be executed.

9.5.6 Shell Code–Based Attacks

Another class of computer worms utilize shell code on the target machine. The basic idea is to run a command prompt on the remote system, such as cmd.exe (on Windows) or /bin/sh (on UNIX) via the exploit code. Consider Figure 9.10 for an illustration.

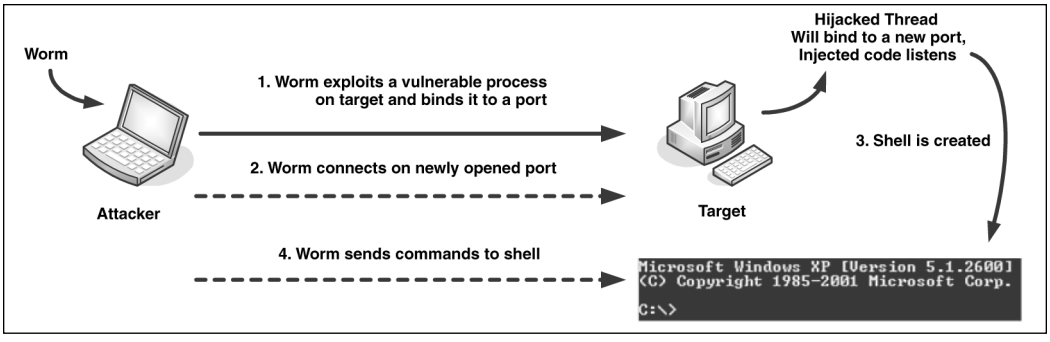

Figure 9.10 A typical shell code–based attack.

The worm follows these steps:

1. It injects code into a remote process and binds a specific port to the process. The exploited process starts to listen on the port.
2. The worm attempts to connect to the listening port.
3. If the connection to the port is successful, the previously injected shellcode executes a command prompt and binds that process to the same port that the attacker is using.
4. Finally, the worm can start to send commands to the shell.

An example of such a worm is W32/Blaster.

Shellcode-based attacks are typically more common on UNIX systems than on Windows systems. A few variations exist, such as back-connecting shellcode and shellcode that reuses an existing connection.

Back-connecting shellcode immediately attempts to connect the target with the attacker by establishing a TCP connection from the target to the attacker's machine. The advantage of this method is that it allows machines behind a firewall to "connect-out" to the attacker system.

This attack requires the attacker system to listen on a particular port and wait for the shellcode to connect, as shown in Figure 9.11.

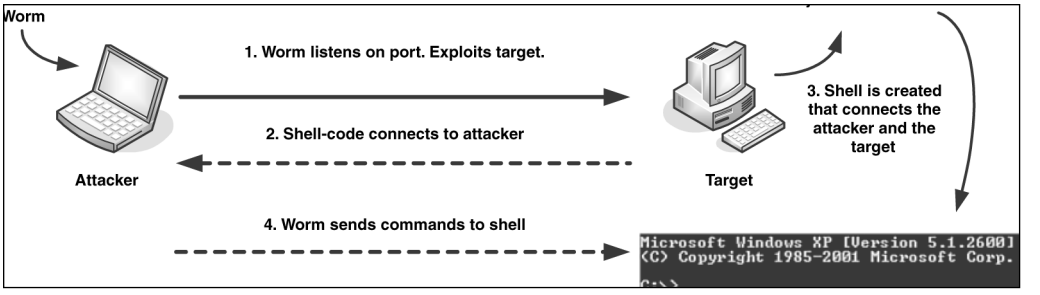

Figure 9.11 A back-connecting shellcode.

The basic difference occurs in the second step. The shellcode executes on the target and connects to the attacker. When the connection is established, the shell-code creates a shell prompt that gets its input from the attacker. The W32/Welchia worm uses this approach.

The exploiting phase might take place in a few steps. For example, Linux/Slapper exploits the target more than once to run shellcode via a heap overflow condition. Slapper, however, implements yet another shell-code technique, reusing the connection established between the attacker's machine and the target. As shown in Figure 9.12, the shellcode does not need to reconnect to the target. In Chapter 15 you can find a traced shellcode of Slapper that illustrates the reused connection better.

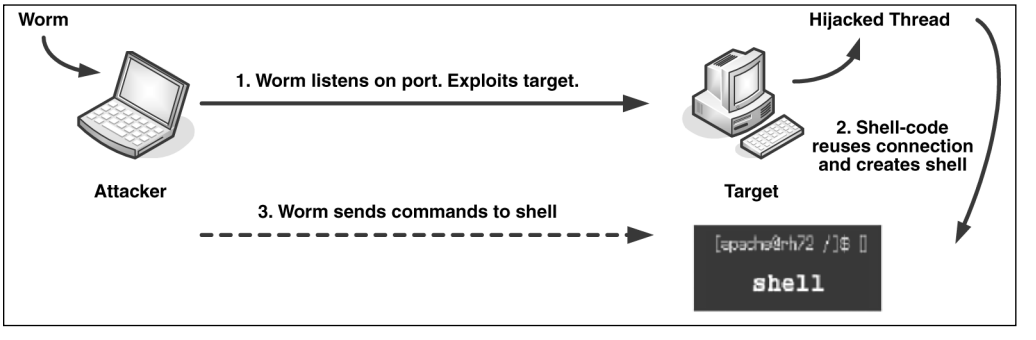

Figure 9.12 A connection-reusing shellcode.

9.6 Update Strategies of Computer Worms

Computer worms can be classified according to their update strategies. An early example of this is W95/Babylonia, a Windows Help and PE infector and self-mailer that was discovered on December 6, 1999.

Babylonia was posted to the alt.crackers Internet newsgroup as a Windows Help file named serialz.hlp[31], which appeared to be a list of serial numbers for commercial software. This Help file was launched by many people who activated the virus on their systems. When executed, the virus creates a downloader component that looks for updates on a Web site. (Figure 9.13 illustrates this.)

First, the downloader reads the content of a text file called virus.txt stored on the Web site. This text file lists a few filenames, such as dropper.dat, greetz.dat, ircworm.dat, and poll.dat. These files use a special plug-in file format with a header that starts with the identifier VMOD (which stands for *virus module*). The header of the virus modules contains an entry point of the module and, using this information, the downloader component of Babylonia downloads and executes the plug-in modules inside its own process, one by one.

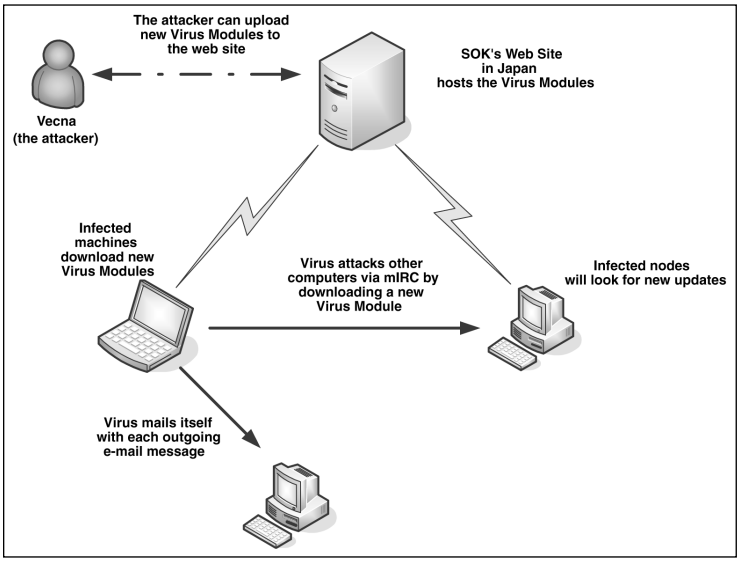

Figure 9.13 The update procedure of Babylonia.

- The dropper.dat module can reinstall the virus code on the system. This can be used by the attacker to update the virus with a newer release or to reinfect an already cleaned system via the downloader.

- The greetz.dat module is the payload. It modifies the c:\autoexec.bat file to display a message, shown in Listing 9.7, in January of each year.
- The ircworm.dat module is an mIRC worm installer that infects other targets via an mIRC.
- The poll.dat module is used to track the number of infected machines. When it is used, it sends messages to babylonia_counter@hotmail.com, with the Portuguese message "Quando o mestre chegara?" ("When will the master arrive?")

Listing 9.7

The Babylonia Worm's Message

```
W95/Babylonia by Vecna (c) 1999
Greetz to RoadKil and VirusBuster
Big thankz to sok4ever webmaster
Abracos pra galera brazuca!!!
---
Eu boto fogo na Babilonia!
```

Not only is Babylonia able to infect two different Windows file formats, it also infects WSOCK32.DLL, allowing it to send e-mails with an attachment whenever the user sends mail. Babylonia somewhat borrows this idea from Happy99.

The weakness of the attack is the update system based on a single Web site. After authorities pulled the site, Babylonia could not download new components.

9.6.1 Authenticated Updates on the Web or Newsgroups

Realizing the weaknesses of a single Web site–based update system, Vecna decided to use alternated update channels and strong cryptography to authenticate the updates. The W95/Hybris worm was released in late 2000. It was an unusually large project of several top virus writers from around the world: Brazilian, Spanish, Russian, and French virus writers were all part of the large team that developed it.

Hybris uses 1,023-bit RSA signing[32] to deliver its update modules to infected systems. It also uses a 128-bit hash function to protect the updates against attacks. The hash function uses XTEA (extended tiny encryption algorithm, which is a successor of TEA). XTEA is in the public domain, written by David Wheeler and Roger Needham. The RSA library for Hybris was written by the infamous Russian virus writer, Zombie. Figure 9.14 is an illustration of the Hybris attack.

Note the interesting selection for XTEA instead of TEA, which was previously found weak by cryptographers John Kelsey, Bruce Schneier, and David Wagner many years ago at CRYPTO 1996. In fact, TEA was used as a hash function in the security of the second version of the Microsoft Xbox. This weakness was leveraged a day later after its announcement by a team headed by Andy Green to break the security of the Xbox scheme by flipping bits in Xbox's FLASH ROM code that allowed a jump instruction to branch to RAM[33].

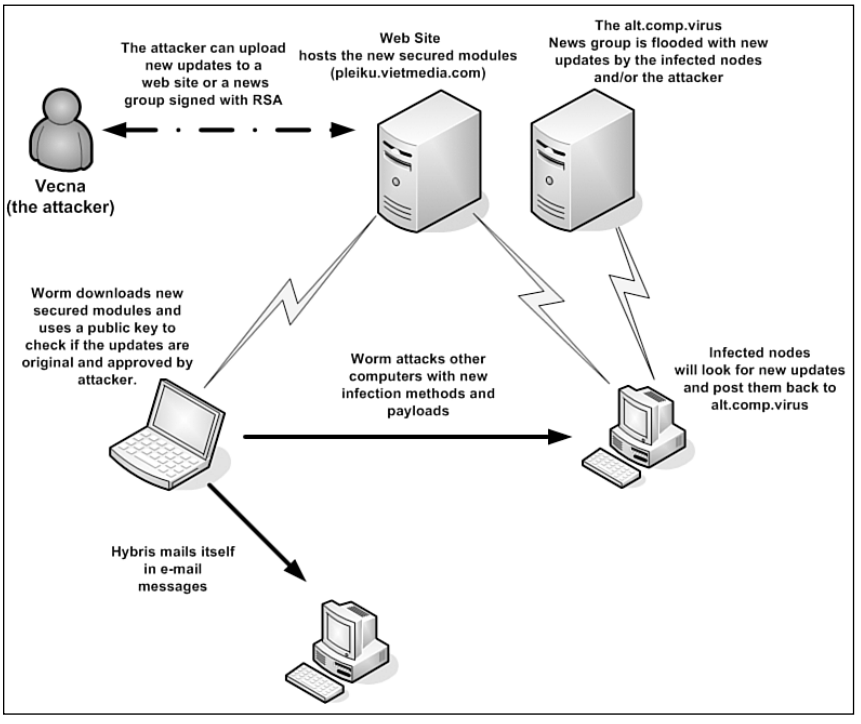

Figure 9.14 The authenticated updates model of the Hybris worm.

The idea of the Hybris worm is to encrypt the updates with XTEA and sign the update files with RSA on the attacker's system. The attacker creates a secret key and a corresponding public key. He puts the public key into the virus, and the XTEA encryption/decryption keys are delivered with the module—but are signed with a 1,023-bit RSA secret key. This is called a hybrid signing technique, which makes the process more efficient.

Instead of using a single 128-bit key, Hybris uses 8 XTEA keys, one of which is a hash computed about the plug-in and 7 other 128-bit keys that are set randomly. First, a 128-bit hash of the module is calculated using XTEA. This value will be

used as one of eight 128-bit encryption keys to encrypt the entire module using a 64-bit XTEA block cipher. The block cipher applies the eight 128-bit keys (including the hash of the plug-in) to each consecutive 64-bit block of the plug-in. Each 64-bit block is encrypted with one 128-bit key. Thus the first 64-bit block is encrypted with the first key in the set, the second 64-bit block is encrypted with the second 128-bit key (the hash) until the keys wrap around: The 9th block is encrypted with the first key again, and so on.

Signing allows the worm instances to check if the update files were distributed by the virus writer. Thus the RSA algorithm is used to prevent changes to plug-ins or to create new plug-ins without specifically involving the attacker who holds the secret key. The worm uses the public key corresponding to the secret key of the attacker to validate the signed XTEA key and verifies that the hash is correct to avoid forgery attacks.

Although the updates are encrypted, the algorithm uses a symmetric key so the modules can be decrypted by anyone, in the same way as the worm decrypts them. The attacker is protected against any manipulations that could occur to update modules. Thus it is not feasible to distribute an update that could kill the worm without the secret key of the virus author unless, of course, there is some implementation error discovered that commonly occurs in cryptography.

There were up to 20 known modules (so-called Muazzins) for Hybris. However, there were more than 32 different versions of these in circulation. After encrypting and signing the module, the attacker encoded the module to send it to the alt.comp.virus newsgroup. Infected systems, which were all looking for the modules, downloaded and decrypted them using their public keys.

Although the initial update Web site was quickly disabled, the attacker had the opportunity to send out new updates in newsgroups. Infected nodes propagated the modules back to the newsgroups, so all infected nodes had a chance to get the updates. Hybris used a similar technique to the Happy99 worm's algorithm to inject its code into the WSOCK32.DLL library, propagating itself via e-mail.

The update modules included several extensions to the worms:

- A DOS EXE file infection module.
- A file infection module to attack PE files without changing their size and CRC 16/32/48 checksum. This module used compression to compress the host and filled the module with extra data, using the algorithm of the Russian virus writer, Zhengxi, to make the CRC the same as it was before the infection.
- A wrapper module to encrypt the Hybris-infected WSOCK32.DLL further.

- A Windows Help file infection module. (This module borrows code from W95/Babylonia.)
- A PE file infection module using Zombie's KME polymorphic engine.
- Two archive infection modules to infect RAR, ZIP, and ARJ archives.
- Two different plug-in modules to infect Microsoft Word documents and a third module to infect Microsoft Excel documents.
- A DoS attack module.
- An encrypted dropper generator module.
- An attacker module to infect machines via a SubSeven backdoor.
- A HATE (human-alike text engine) message module; this particular module could generate e-mail messages in the names of well-known antivirus researchers such as Eugene Kaspersky, Mikko Hypponen, and Vesselin Bontchev. My name was also on the list. The module was supposed to send e-mail messages using one of my e-mail addresses in the sender field with the subject "Uglier than Hermann Monster!" (most likely a reference to Herman Munster) with the attachment named "The Hungarian Freak!.exe."

Note:

This module was written by the Spanish virus writer, Mr. Sandman, the founder of the 29A virus writing group, who is believed to be a professional translator. Many other viruses of Mr. Sandman's are related to his interest in languages, for example Esperanto and Haiku.

- A retro attack module to block access to antivirus Web sites.
- Another e-mail message generator using a SOAP Web server to generate fortune cookie messages and send these (with Hybris) to recipients.
- A sys file infection routine to hide the infected WSOCK32.DLL on the system with stealth routines.
- An exploit module that can be used to retrieve files from vulnerable Web servers.
- Another retro attack to scan the disk and Registry for antivirus programs and delete them or corrupt their databases.
- An e-mail-based tracker module to send e-mail messages from infected nodes to a particular e-mail account.

- A few other generic message generator modules for e-mail propagation.
- A Happy 2000 module. This one overwrites the SKA.EXE file of the Happy99 worm to propagate Hybris instead. It also contains the graphical payload of the Happy99 worm.
- A module to download additional plug-in modules from Web sites.
- A Usenet module to connect to NNTP servers and download plug-ins. This module also uploads other modules to a newsgroup.
- Finally, an OpenGL-based animation that installed itself to load at boot time. This module, shown in Figure 9.15, was contributed by the French virus writer, Spanska.

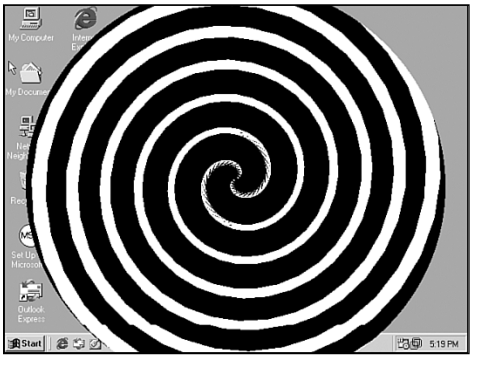

Figure 9.15 The OpenGL-based hypnotizer spiral plug-in.

Listing 9.8 is an example of a plug-in module posted to the alt.comp.virus newsgroup[34].

Listing 9.8

A Hybris Update in alt.comp.virus (Partial Snippet)

```
Date: Tue, 24 Jul 2001 20:29:51 -0700
Newsgroups: alt.comp.virus
Subject: h_2k MRKR KRnAbIvQdE?UlOhK6CrWdU#YvYnM:SrYU

TRUTUWXXPTVFVY3NXSTREYCUSPVNBLZLSQBPXXRRYMUOD7USWESFRWYBUTREMBLWKSPS
OXYVNWZG KTVHVDMTTRODVSMCZFWCQXSXVVTZVUKVKHOBTRNFYVVBLFRBXWUVRHWHPF
SE&THUFNVMHZCRHNVRVZUKXVWSBSBZRPB6NEVVYZLSVSLDLZZFZCYCSWKDLUZVYR5ZYLZ
NDOSNUKRMUYXOHTEMUKD
```

The body of this message contains the Happy 2000 plug-in of Hybris (only a snippet is shown in Listing 9.8). The name of the plug-in is in the Subject line as

"h_2k," which is followed by the version number information of the plug-in. Hybris uses the version information to decide whether a module needs to be extracted and executed.

9.6.2 Backdoor-Based Updates

Several computer worms open up a port on the compromised system and implement an interface to execute arbitrary files on the compromised machine. The attacker can use this interface to update the worm's code from one version to another. For instance, the W32/Mydoom worm opens a TCP port in the range of 3127 to 3198 and waits for a connection, implementing a simple protocol. Essentially, Mydoom's code is updated similarly to a backdoor-based propagation technique described earlier in this chapter. The attacker needs to scan for systems that have a port open and can send an executable to the target that will be executed on the remote node. The first few versions of Mydoom did not implement any security mechanism for their update protocol. Not surprisingly, worms such as W32/Doomjuice, W32/Beagle, and W32/Welchia attacked Mydoom-compromised systems by taking advantage of the insecure update mechanism.

Later releases of Mydoom leave less chance for opportunistic attackers because they inspect incoming requests more carefully.

9.7 Remote Control via Signaling

Attackers often want to control their creations remotely, for example, to execute a DoS attack against a selected target or to control the propagation of the worm to new systems. The most obvious technique is based on the use of a backdoor feature built into the worm that communicates directly with a particular host. However, other known techniques centrally control the worm, such as via IRC or Windows domain mail-slots. Consider Figure 9.16, which illustrates the attacks of W32/Tendoolf.

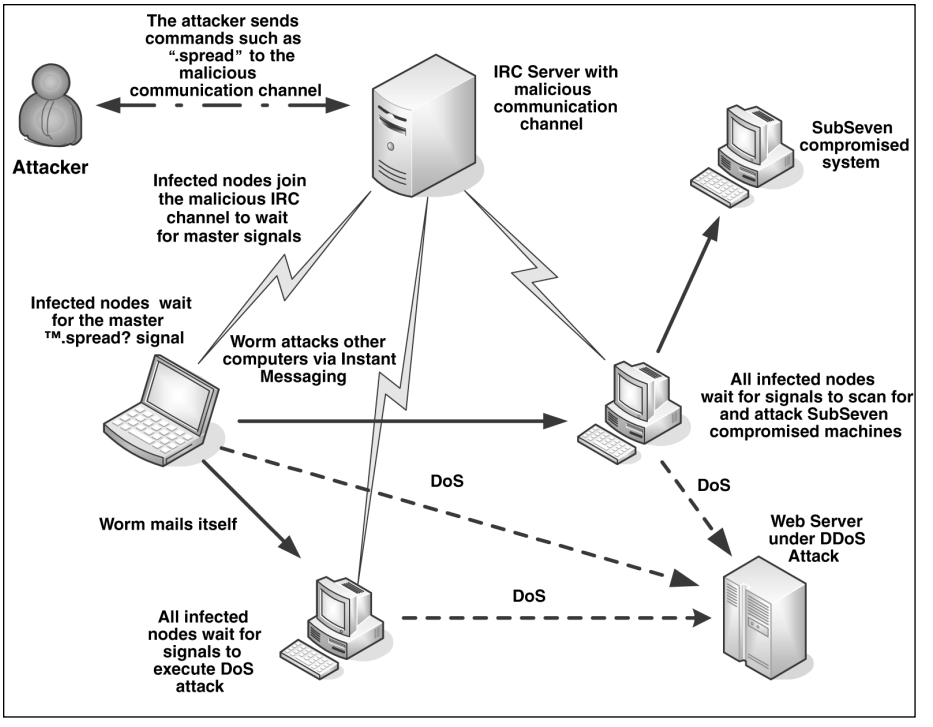

Figure 9.16 The remote-controlled Tendoolf worm.

Some of the variants of this worm only propagate the infection to new targets whenever the attacker sends a ".spread" signal to an IRC server into a discussion channel of the worm itself. When the attacker sends the ".spread" signal, the already compromised systems look for new targets to infect. Techniques include e-mail, instant messaging, and SubSeven backdoor-compromised nodes attack. In addition, the attacker can execute a DoS attack against any target. The target of the DoS attack is unknown to the compromised systems (it is not hard-coded in the worm) until the attacker sends the command. Then the compromised nodes turn against the specified target and execute various kinds of flooding methods. Hence the name of the worm, which is Floodnet spelled backwards.

This kind of remote control–based worm propagation is often confusing for junior antivirus researchers.

9.7.1 Peer-to-Peer Network Control

Some computer worms implement a virtual network between infected nodes to establish communication and control operations. The Linux/Slapper worm is an

example of this kind of computer worm. Slapper uses the UDP protocol and port 2002 on each infected node. When the worm infects a new target, it passes the attacker's IP address to the target system. Then each node receives the IP addresses of all other nodes and keeps these in a list. Whenever a new IP address is introduced, all other nodes receive the update via this special virtual network, which uses TCP-like (stateful) features to ensure that the information arrives at the target correctly. The infected nodes select a random set of other machines to broadcast the updated information on the network. This is called a *broadcast segmentation technique*.

The remote control interface of Linux/Slapper is very advanced. The code is borrowed from the BSD/Scalper worm, which was based on a previous attacker tool. Just like Scalper, Slapper implements a hierarchical network structure[35] that keeps track of systems the worm has infected. Slapper supports a large set of commands that implement UDP flood, TCP SYN flood, IPv6 SYN flood, and DNS standard query flood DoS attacks; it also supports the execution of arbitrary commands on the compromised nodes.

Consider Figure 9.17 for a possible illustration of a Slapper worm infection. When the first node is infected, it receives the IP address of the attacker host, followed by the list of all other nodes that are in the network, and so on.

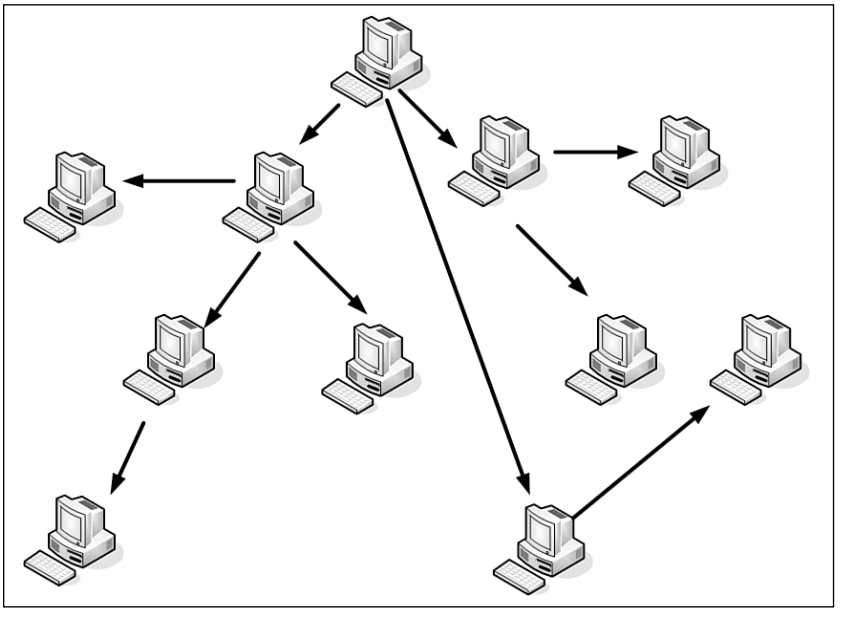

Figure 9.17 A Slapper worm infection.

Figure 9.17 only illustrates how the infection is passed from one system to the next. Figure 9.18 is a possible way to illustrate the hierarchical relationships between the infected nodes as a P2P command network. Because more than one initial infection might be started by the attacker, there might be multiple smaller and larger P2P networks parallel to each other without any connection, similar to the configuration shown in Figure 9.18.

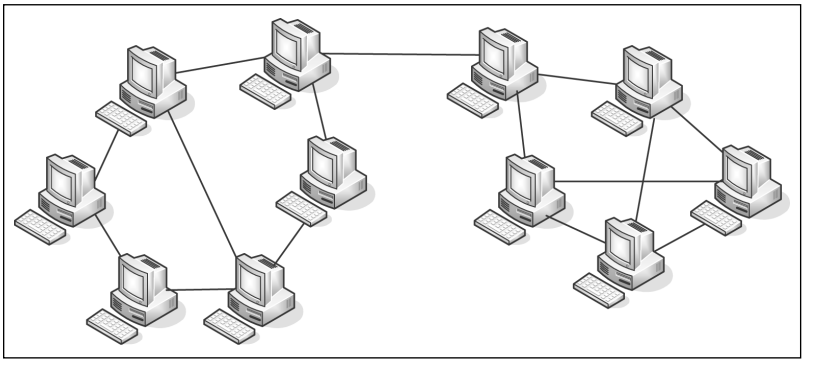

Figure 9.18 Slapper worm P2P network hierarchy.

9.8 Intentional and Accidental Interactions

Computer virus researchers have observed a set of interesting behaviors that are the result of intentional and accidental interactions between various kinds of malicious code. This section describes common interactions.

9.8.1 Cooperation

Some computer viruses accidentally cooperate with other malicious code. For example, computer worms might get infected with a standard file infector virus as they pass through already infected nodes. It is common to encounter multiple infections on top of in-the-wild computer worms. It is not uncommon to find three or even more different viruses on the top of a worm carrier. This can help both the network worm and the standard file infector virus in various ways.

A worm can take advantage of the infection of an unknown file infector virus. If the file infector virus is unknown to antivirus products, the computer worm body might not be detectable. For example, in some cases the worm body will be embedded deep inside the virus code, leaving little chance for the antivirus program to find it. See Figure 9.19 for an illustration.

In Step A, the computer becomes infected with a worm. In Step B, the worm successfully penetrates a new remote system. That computer, however, is already infected with a virus that infects exactly the same type of files in which the worm propagates itself. Thus the file infector virus attaches itself to the worm. In Step C, the multiple infection arrives at a new computer. When the worm is executed, the file infector virus runs (in most cases, it will run before the worm's code) and infect other objects.

In Step D, the combination arrives at a system protected by antivirus software that knows the file infector virus on top of the computer worm carrier—but does not know the computer worm underneath. If the antivirus can disinfect the file infector virus, it might create a file object that is not exactly the same as the original worm. For example, the binary file of the worm might get larger or smaller, and important fields in its header might also change. (Of course, an antivirus is just one possible agent that might interact with other malicious programs.)

Thus a "mutant" worm will have the chance to propagate itself further and infect a new system in Step E. In practice, no antivirus software would consider the Step E case a variant of the original worm. However, antivirus programs need to address this issue. For example, the MD5 checksums of the "mutant" worm body are clearly different, and if the antivirus or content-filtering software uses such checksums to detect the original worm, it will fail to detect the "mutant" worm.

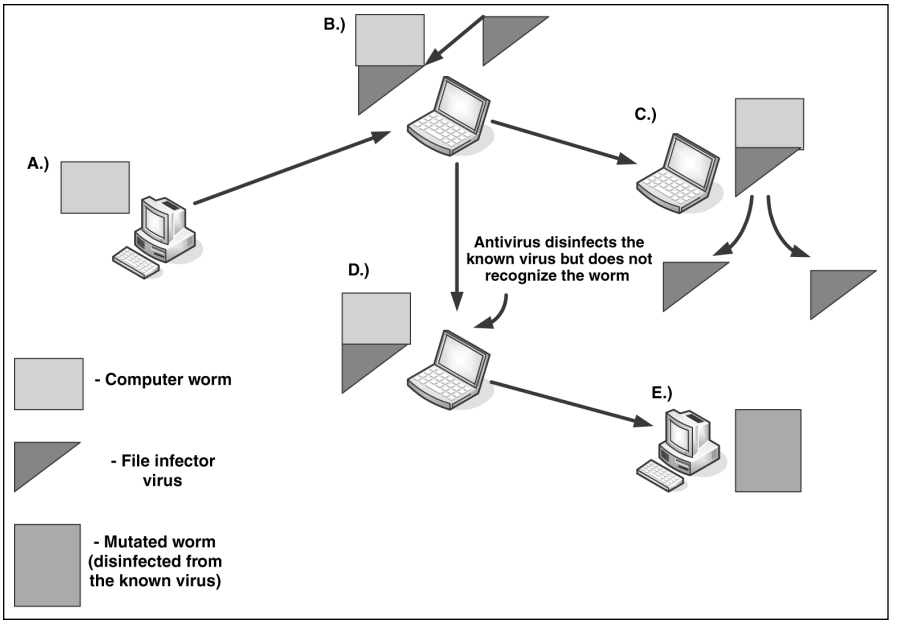

Figure 9.19 The accidental interaction of a worm infected with a file infector virus.

An intentional variation of such cooperation appeared in the "symbiosis project" of GriYo. He released the mass-mailing worm, W32/Cholera, in infected form. The worm was infected with the polymorphic W32/CTX virus and became a carrier. The result was a quick succession of both W32/Cholera and W32/CTX infections worldwide. As a result, W32/CTX was reported on the Wildlist.

File infector viruses such as W32/Funlove often infect other worms and often multiple times. Such file infector viruses occasionally disappear from the top 50 charts but suddenly attack again during major computer worm outbreaks. This kind of activity was seen increasingly when the W32/Beagle worm appeared in the wild. As discussed earlier, some variants of Beagle send a password-protected attachment to recipients. Because the worm creates the archive (ZIP) file on the local system, the executable file of the worm can easily get infected by viruses such as Funlove before the executable gets packed. Thus a virus like Funlove also enjoys being password-protected[36]. Because several antivirus softwares had problems detecting the password-protected attachments reliably, "traveler viruses" could take advantage of this accidental cooperation.

A form of cooperation also exists with the previously mentioned W32/Borm creation, which infects Back–Orifice-compromised systems. W32/Borm does not attempt to kill Back Orifice; it simply takes advantage of compromised systems to

propagate. Similarly, the aforementioned Mydoom had a backdoor that was utilized by the Doomjuice worm to spread itself.

With macro and script viruses, "body snatching" attacks often occur. Two or more script or macro viruses might form a new creation as they accidentally propagate each other's code.

9.8.2 Competition

Competition between malicious codes was also experienced among computer viruses. Several viruses attack other viruses and disinfect them from the systems that they have compromised. An example of this is the Den_Zuko boot virus[37], which disinfects the Brain virus. These viruses are often called "benefical viruses" or "antivirus" viruses.

Antiworm computer worms started to become more popular in 2001 with the appearance of the CodeRed worm and the counterattacking, CodeGreen. (However, antiworm worms had been experienced previously on other platforms, such as Linux.)

Because IIS could be exploited more than once, CodeGreen could easily attack CodeRed-infected systems. The worm sent a similarly malformed GET request to the remote target nodes to CodeRed, which had in front the message shown in Listing 9.9.

Listing 9.9

The Front of a CodeGreen GET Request

```
GET /default.ida?Code_Green_<I_like_the_colour-_-><AntiCodeRed-
CodeRedIII-IDQ_Patcher>_V1.0_beta_written_by_'Der_HexXer'-
Wuerzburg_Germany-_is_dedicated_to_my_sisterli_'Doro'.
Save_Whale_and_visit_<www.buhaboard.de>_and_www.buha-security.de
```

The worm also carried the following messages shown in Listing 9.10.

Listing 9.10

Other Messages of the CodeGreen Worm

```
HexXer's CodeGreen V1.0 beta CodeGreen has entered your system
it tried to patch your system and
to remove CodeRedII's backdoors

You may uninstall the patch via
SystemPanel/Sofware: Windows 2000 Hotfix [Q300972]

get details at "www.microsoft.com".
visit "www.buha-security.de
```

CodeGreen removed the CodeRed infections from systems and also removed the backdoor components of other CodeRed variants. Furthermore, it downloaded and installed patches to close the vulnerability.

Similar attacks against W32/Blaster worm were experienced when the W32/Welchia worm began its antiworm hunt against Blaster, which started the "worm wars" (as I decided to call it after Core Wars).

Another enthralling example is the W32/Sasser worm. Sasser targeted an LSASS vulnerability that was previously exploited by variants of Gaobot worm. Thus Gaobot's author was not impressed because Gaobot needed to compete with Sasser for the same targets. Consequently, the W32/Gaobot.AJS[38] worm was developed with a vampire attack. I decided to call this kind of attack a "vampire" based on the Core War vampire attack. Vampire warriors can steal their enemies' souls (see Chapter 1, "Introduction to the Games of Nature," for details).

Gaobot.AJS is a vampire because it attacks Sasser when the two worms are on the same machine. Instead of simply killing Sasser, Gaobot.AJS modifies Sasser's code in a very tricky way. As a result of the modification, Sasser can still scan for new targets and even exploit them successfully. However, when Sasser connects to its shellcode on the compromised system to instruct it to download and execute a copy of Sasser via FTP, the code modifications of Gaobot.AJS will get control. In turn, Gaobot.AJS sends commands to Sasser's shellcode on the remote machine and instructs it to download a copy of Gaobot.AJS's code instead of Sasser's. Furthermore, Gaobot closes the connection to the remote machine so Sasser cannot propagate but is used as a Gaobot propagation agent in a parasitic manner.

Another gripping example is the W32/Dabber worm, which appeared right after Sasser. As mentioned, Sasser's shellcode is instructed to download a copy of Sasser via FTP. On the attacker system, Sasser implements a crude FTP server. However, this routine of Sasser had a simple buffer overflow vulnerability that could be exploited. (Indeed, worms can have their own vulnerabilities!) Dabber was released to exploit Sasser's vulnerability to propagate itself. It scans for targets that were compromised by Sasser and attempts to connect to Sasser's vulnerable "FTP server" to exploit it successfully.

It is expected that competitions between malicious programs will become more and more common in the future.

9.8.3 The Future: A Simple Worm Communication Protocol?

Although increased competition among malicious programs is likely, it also makes sense for attackers to invest in cooperating techniques. For example, computer worms could use a special protocol such as simple worm communication protocol (SWCP) to exchange information, as well as plug-ins ("genes") among different families of computer worms that support SWCP. Computer worms could swap payloads, exchange information about systems to attack, or even collect e-mail addresses and share them with the other worms that occasionally communicate using SWCP. I highly anticipate that such techniques will appear in the very near future.

Of course, communication can have other forms. For instance, viruses could "reproduce sexually"[39] to cross their genomes to produce offspring, which can evolve or devolve. The closest currently known example of accidentally "sexually reproducing" computer viruses can be found in macro viruses which occasionally swap, or snatch their macros ("genes") as discussed in Chapter 3, "Malicious Code Environments,". However, specifically written binary viruses could possibly demonstrate similar behavior that would lead to further evolution in computer viruses on their own.

9.9 Wireless Mobile Worms

The SymbOS/Cabir worm[40] indicates a totally new era of computer worms that will slowly become more popular as wireless smart phones replace current mobile phone systems, which have limited programming ability. The Cabir worm appeared in June 2004, and it has a number of unique features. This worm can run on Nokia 60 series phones running the Symbian operating system. The Symbian operating system is based on the EPOC. In fact, Symbian is EPOC version 6, also called EPOC32, but has a new name.

Interestingly, the Cabir worm spreads using the Bluetooth feature of wireless phones as shown in Figure 9.20.

Figure 9.20 The attacker phone is on the left, and the recipient is on the right.

The worm's code is compatible with mobile phones using ARM series processors with Symbian operating system. Normally, by default the Bluetooth communication feature is off on mobile phones. Mobile phone users might exchange some little programs, and in doing so they open up the Bluetooth communication channel to Cabir-like worms as well.

When executed, Cabir installs itself into several directories of the Symbian OS intending to make sure it will run each time the user boots the phone. Fortunately, this operation is disallowed in newer phone models. However, on older phones, worm components cannot be easily found without using custom file manager applications. Cabir does not enumerate Bluetooth devices; instead, it tries to find only the first such device and communicates with that device. The standard Bluetooth range is about 30 feet, and apparently not all Bluetooth devices like to communicate with each other. (However, researchers such as Mark Rowe are experienced with Bluetooth signal amplification and pointed out that attackers could utilize such technology to extend the Bluetooth range to about 300 feet, reliably.) In addition, researchers such as Ollie Whitehouse of @stake also demonstrated that Bluetooth devices are discoverable even in the so-called "non-discoverable" mode[41]. Several Bluetooth-related attack tools exist today including the most popular Bluesniff, Btscanner, PSMscan, and Redfang.

During the natural infection tests, Cabir first talked to a Bluetooth printer, which strangely acted as a "sticky" honeypot system and blocked the worm given that the printer did not support the Object Exchange (OBEX) protocol that is required to send a file. However, the worm successfully infected another phone as soon as I turned the Bluetooth printer off. Cabir is overly active in finding other phones and that can easily drain the battery of the phone similarly to natural situations when your phone is hopelessly attempting to find a provider without finding one in range.

A further problem is that you need to "hide" with mobile phones when you test replicate worms. Although the recipient needs to accept the incoming message to successfully receive the message, you do not want to infect another phone "by accident." In fact, there are several known vulnerabilities of Bluetooth systems, and some of these can be utilized to execute arbitrary code on Pocket PC devices[42], while others can be used to implement phishing attacks on a number of smart phones types[43].

Sure enough, in the future you can expect that worms are going to make phone calls from your mobile phone instead of you. There might be a new era of MMS- (Multimedia Messaging Service) based mass mailer worms as well as SMS- (Short Messages Services) based downloaders, porn dialers, and spammer applications, as well. Who is going to pay the bill?

References

1. Dr. Vesselin Bontchev, personal communication, 2004.
2. Nick FitzGerald, "When Love Came to Town," *Virus Bulletin*, June 2000, pp. 6-7.
3. Donn Seeley, "A Tour of the Worm," *USENIX Conference*, 1989, pp. 287-304.
4. Frederic Perriot and Peter Szor, "An Analysis of the Slapper Worm Exploit," *Symantec Security Response, White Paper*, April 2003, `www.sarc.com/avcenter/whitepapers.html`.
5. "The Cheese Worm," CERT Incident Note IN-2001-05, `http://www.cert.org/incident_notes/IN-2001-05.html`.
6. "The sadmind/IIS worm," *CERT Advisory CA-2001-11*, `http://www.cert.org/advisory/CA-2001-11.html`.
7. Alexsander Czarnowski, "Distributed DoS Attacks—Is the AV Industry Ready?" *Virus Bulletin Conference*, 2000, pp. 133-142.
8. Ido Dubrawsky, "Effects of Worms on Internet Routing Stability," Security Focus, June 2003, `http://www.securityfocus.com/infocus/1702`.

9. Frederic Perriot, "Crack Addict," *Virus Bulletin*, December 2002, pp. 6-7, http://www.virusbtn.com/resources/viruses/indepth/opaserv.xml.

10. National Bureau of Standards, "Data Encryption Standard," *FIPS Publication 46*, U.S. Department of Commerce, 1977.

11. Electronic Frontier Foundation, "Cracking DES," Sebastopol, CA, 1998, ISBN: 1-56592-520-3 (Paperback).

12. Dr. Steve R. White, "Covert Distributed Processing with Computer Viruses," *Advances in Cryptology—CRYPTO '89*, Springer-Verlag, 1990, pp. 616-619.

13. Vesselin Bontchev, "Anatomy of a Virus Epidemic," *Virus Bulletin Conference*, 2001, pp. 389-406.

14. Eugene H. Spafford, "The Internet Worm Program: An Analysis," 1988.

15. Katrin Tocheva, "Worming the Internet—Part 2," *Virus Bulletin*, November 2001, pp. 12-13.

16. Peter Ferrie, "Sleep-Inducing," *Virus Bulletin*, April 2003, pp. 5-6.

17. Katrin Tocheva, Mikko Hypponen, and Sami Rautiainen, "Melissa," March 1999, http://www.f-secure.com/v-descs/melissa.shtml.

18. Peter Ferrie, "Magisterium Abraxas," *Virus Bulletin*, May 2001, pp. 6-7.

19. Gabor Szappanos and Tibor Marticsek, "Patriot Games," *Virus Bulletin*, July 2004, pp. 6-9.

20. Peter Ferrie and Peter Szor, "Sircamstantial Evidence," *Virus Bulletin*, September 2001, pp. 8-10.

21. Dmitry O. Gryaznov, "Virus Patrol: Five Years of Scanning the Usenet," *Virus Bulletin Conference 2002*, pp. 195-198.

22. Peter Szor, "Parvo—One Sick Puppy?" *Virus Bulletin*, January 1999, pp. 7-9.

23. Atli Gudmundsson and Andre Post, "W32.Toal.A@mm," http://securityresponse.symantec.com/avcenter/venc/data/w32.toal.a@mm.html.

24. Peter Szor, "Happy Gets Lucky?" *Virus Bulletin*, April 1999, pp. 6-7.

25. Stuart McClure, Joel Scambray, and George Kurtz, "Hacking Exposed: Network Security Secrets and Solutions," 3rd Edition, Osborn/McGraw-Hill, Berkeley, 2001, ISBN: 0-07-219381-6 (Paperback).

26. Vern Paxson, Stuart Staniford, and Nicholas Weaver, "How to 0wn the Internet in Your Spare Time," http://www.icir.org/vern/papers/cdc-usenix-sec02/.

27. Neal Hindocha and Eric Chien, "Malicious Threats and Vulnerabilities in Instant Messaging," *Symantec Security Response, White Paper*, October 2003, www.sarc.com/avcenter/whitepapers.html.

28. "Buffer Overflow in AOL Instant Messenger," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0005.

29. Sergei Shevchenko, "W32.Taripox.A@mm," February 2002, http://securityresponse.symantec.com/avcenter/venc/data/w32.taripox@mm.html.

30. Katrin Tocheva and Erdelyi Gergely, "Aplore," April 2002, http://www.f-secure.com/v-descs/aplore.shtml.

31. Marious van Oers, "Digital Rivers of Babylonia," *Virus Bulletin*, February 2000, pp. 6-7.

32. Ronald L. Rivest, Adi Shamir, and Leonard Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, v-21, n-2, February 1978, pp. 120-126.

33. Andrew "bunnie" Huang, "Hacking the Xbox," *Xenatera LLX*, San Francisco, 2003, ISBN: 1-59327-029-1.

34. Nick Fitzgerald, personal communication, 2001.

35. Sen Hittel, "Modap OpenSSL Worm Analysis," *Security Focus*, September 16, 2002.

36. Dr. Igor Muttik, personal communication, 2004.

37. Vesselin Bontchev, "Are 'Good' Computer Viruses Still a Bad Idea?" *EICAR*, 1994, pp. 25-47.

38. Heather Shannon, Symantec Security Response, personal communication, 2004.

39. Edward Fredkin, "On the Soul," 2000, Draft Paper, http://www.digitalphilosophy.org/on_the_soul.htm.

40. Peter Ferrie and Peter Szor, "Cabirn Fever," *Virus Bulletin*, August 2004, pp. 4-5, http://pferrie.tripod.com/vb/cabir.pdf.

41. Ollie Whitehouse, "Redfang: The Bluetooth Device Hunter," 2003.

42. "WIDCOMM Bluetooth Communication Software Multiple Buffer Overflow Vulnerabilities," http://www.securityfocus.com/bin/10914/discussion.

43. "Bluetooth Information Disclosure Vulnerability," http://www.securityfocus.com/bin/9024/discussion.